LA-UR-14-26124
Approved for public release; distribution is unlimited.

Title: Non-LTE Opacity Computation on GPUs

Author(s): Holladay, Daniel

Intended for: Talk

Issued: 2014-08-04

Non-LTE Opacity Computation on GPUs

Daniel A. Holladay

Thursday 31st July, 2014

Mentors: John G. Wöhlbier (CCS-2) and Chris J. Fontes (XCP-5)

Outline

Non-LTE

Solution Method

Results

Current Status and Looking Forward

Acknowledgements

A big thanks to:

John Wöhlbier and Chris Fontes for providing nearly daily guidance in this work.

Motivation

- In-line non-LTE capability currently exists in xRage, but is slow and uses highly averaged atomic physics models to lower compute times.
- Want to model high energy density physical systems with more complete models and in less time
- Many physical systems of interest are not in local thermodynamic equilibrium (LTE)
  - Gold wall absorption in ICF hohlraums
  - Z machine: existing iron experiment cannot be explained. Non-LTE effects?
  - Star coronae

The Goal

- To produce a robust in-line code capable of computing reasonably accurate opacities with the assumption of LTE relaxed (non-LTE).
  - Near term: demonstrate acceleration of non-LTE opacity computation.
  - Far term (if funded): connect to application codes with in-line capability and compute opacities.
- Use efficient algorithms that expose many levels of parallelism and utilize good memory access patterns for use on advanced architectures.
- Portability to multiple types of hardware including multicore processors, graphics processing units (GPUs), and many core coprocessors such as the intel Xeon Phi and Knights Landing that will be in Trinity (2016).
- Easily coupled to radiation hydrodynamics and thermal radiative transfer codes.

High Energy Density Physics Nomenclature

- The opacity
  - $\rho\kappa_{tot}\left(\mathbf{r}, E = h\nu, t\right)\mathrm{d}s =$ the probability that a photon of energy $h\nu$ will have an interaction in path length $\mathrm{d}s$ about position $\mathbf{r}$ at time t.
- The emissivity
  - $\varepsilon\left(\mathbf{r}, \mathbf{\Omega}, E = h\nu, t\right)\mathrm{d}\Omega\mathrm{d}^3 r\mathrm{d}(h\nu)\mathrm{d}t \equiv$ the expected number of photons emitted in volume $\mathrm{d}^3 r$ about position $\mathbf{r}$ in directions $\mathrm{d}\Omega$ about $\mathbf{\Omega}$ with energies in $\mathrm{d}(h\nu)$ about $h\nu$ in time $\mathrm{d}t$ about t.
- In LTE the emissivity is a product of the absorption cross section and the Planck distribution, $\varepsilon = \rho\kappa_{abs}(\rho, h\nu, T)B(h\nu, T)$
  - Does not hold in non-LTE.
  - Emissivity must be computed independently from the opacity in that case.

Computing an Opacity

What is needed?

- Material mass density, ρ
- Free electron number density, N_e
- Number density of atoms in every populated ionization stage $i \in (1,...,I)$ and every energy level $\ell \in (1,...,L_i)$, $N_{i\ell}$
- In general: radiation, electron, and ion energy distributions. Typically specified via temperatures, T_e (Maxwell-Boltzmann) and T_r (Planck).

$$\kappa_{tot}\left(\rho, T_e, T_r, h\nu\right) = \sum_{i=1}^{I}\sum_{\ell=1}^{L_i} \frac{N_{i\ell}(\rho, T_e, T_r)}{\rho}\left(\sigma_{i\ell}^{(\text{bound–bound})}(h\nu) + \sigma_{i\ell}^{(\text{bound–free})}(h\nu)\right)$$

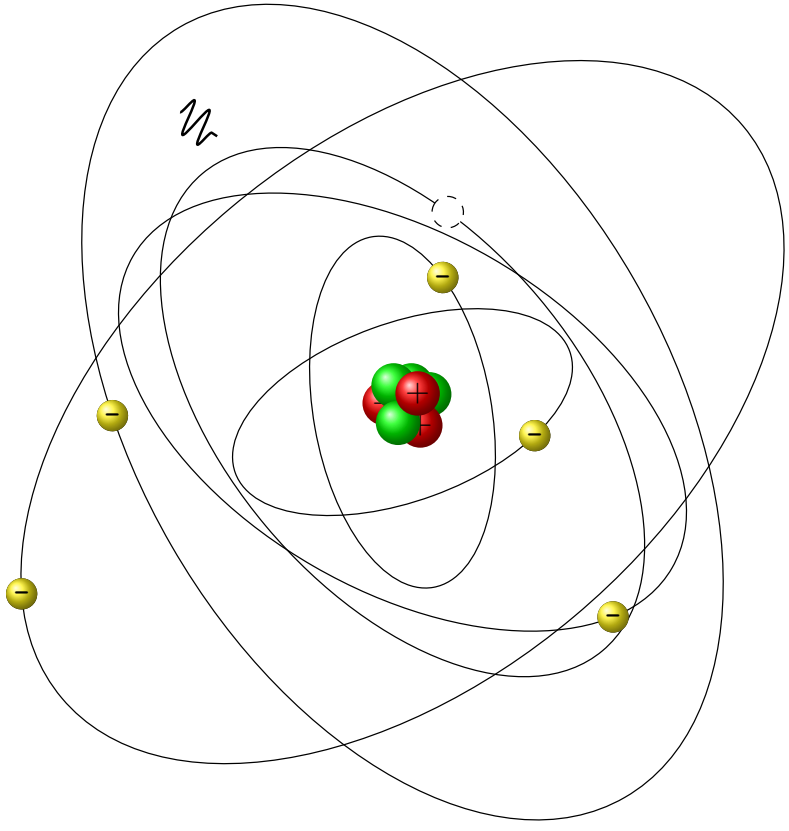
$$+\frac{N_e}{\rho}\int_{-1}^{1}\sigma_s^{(\text{free–free})}\left(\mu, h\nu\right)\mathrm{d}\mu + \kappa^{(\text{free-free})}(h\nu)$$

Atomic Processes

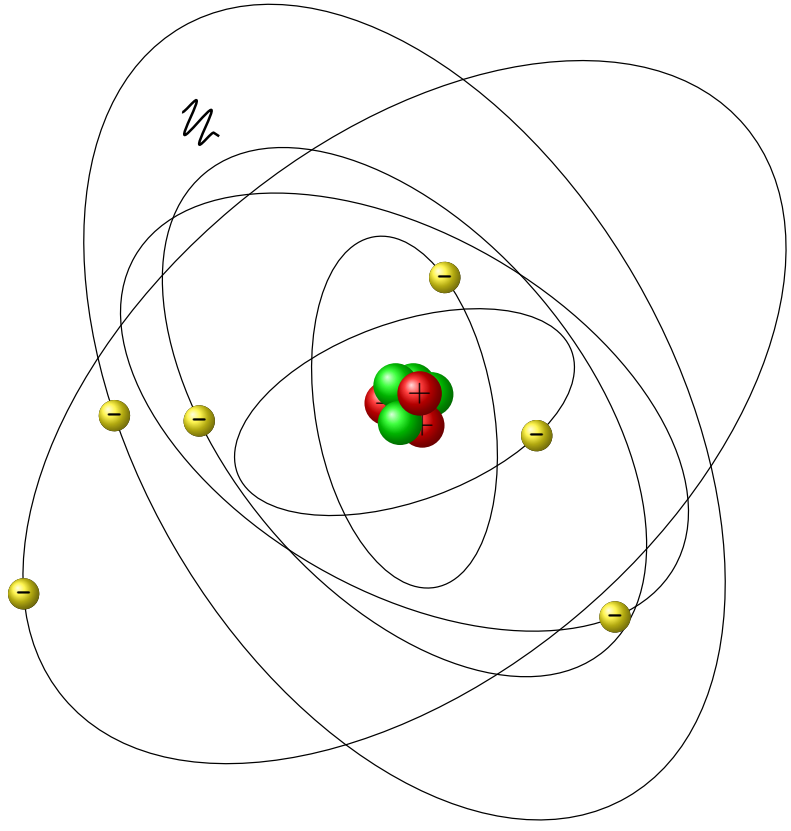
When computing the $N_{i\ell}$'s, the processes by which an atom in a given level ℓ of a given ion stage i can change to another level ℓ', in possibly a different ion stage i', must be identified.

- Atomic Processes
  - Radiative excitation
    - Inverse Process: Radiative de-excitation
  - Collisional excitation
    - Inverse Process: Collisional de-excitation
  - Radiative ionization
    - Inverse Process: Radiative recombination
  - Collisional ionization
    - Inverse Process: 3-body recombination
  - Auger ionization
    - Inverse Process: Dielectronic recombination

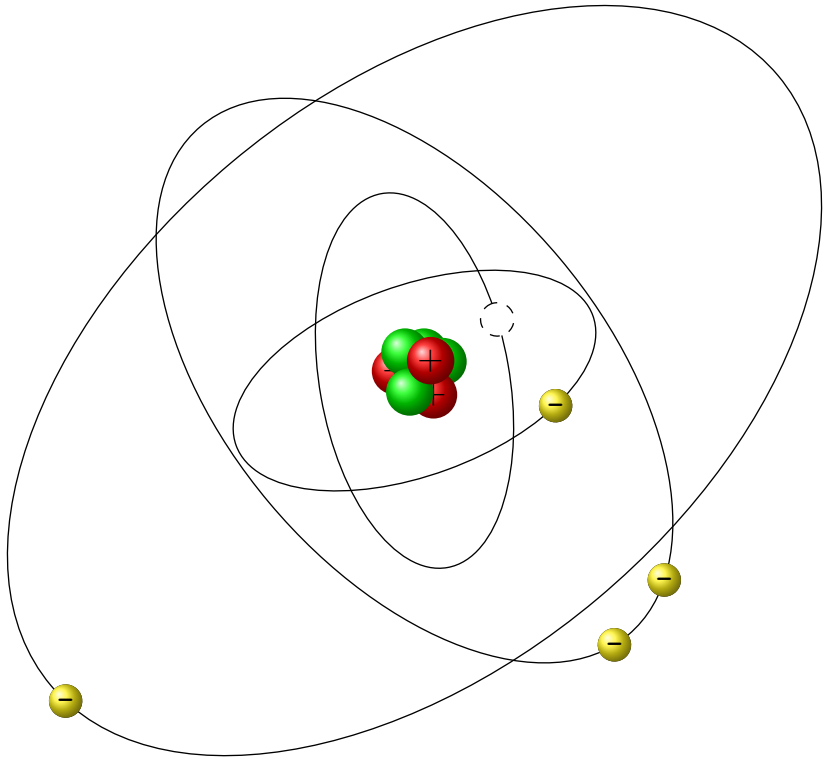
Radiative and Collisional Excitation/De-excitation



Radiative and Collisional Ionization/Recombination



Auger Ionization/Dielectronic Recombination



The Rate Equations

The rates of these processes can be used to solve for the populations for each ion and energy level. These rates form a rate matrix:

$$\frac{\mathrm{d}\mathbf{N}_{i\ell}(N_e)}{\mathrm{d}t} = \Big[\underbrace{C_{i\ell}(N_e)}_{\text{Collisional Ionization}} + \underbrace{P_{i\ell}(N_e)}_{\text{Radiative Ionization}} + \underbrace{A_{i\ell}(N_e)}_{\text{Auger Ionization}} + \underbrace{S_{i\ell}(N_e)}_{\text{Collisional Excitation}} + \underbrace{U_{i\ell}}_{\text{Radiative Excitation}} \Big] \mathbf{N}_{i\ell}$$

- Matrix elements themselves nonlinear functions of N_e.
- We assume steady state ($\frac{\mathrm{d}\mathbf{N}_{i\ell}}{\mathrm{d}t} = \mathbf{0}$) case.

Rate Matrix Properties

- Block tridiagonal
  - Non-uniform block row dimensions
  - leads to rectangular off-diagonal blocks
- Number of block rows:
  - Maximum: Z+1 (extra for fully stripped stage), negative ions not considered
  - Windowing: look at a small of ion stages (not currently implemented)
- Number of levels per stage:
  - Varies based upon model complexity
  - Complex: detailed configuration accounting (DCA) ($\sim O(100) - O(10^6)$ levels per stage)
  - Simpler: r(educed)DCA ($\sim O(10) - O(100)$ levels per stage)
- Matrix elements depend on the unknown $N_e \rightarrow$ matrix elements need to be recomputed and the linear system solved and resolved *multiple times in every non-LTE cell at every time-step.*
- Example cases:
  - Hydrogen has ~ 10 unknowns
  - Iron has ~ 750 unknowns
  - Gold has $\sim 10^4$ unknowns

Cyclic Reduction Algorithm

We start with a block tridiagonal linear system of the form:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 \\ 0 & \cdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & \cdots & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix}$$

Cyclic Reduction Algorithm

Now add a column on both ends of the matrix and let $a_1 = c_n = 0$ and $x_0 = x_{n+1} = 0$.

$$\begin{bmatrix} a_1 & b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_3 & b_3 & c_3 & 0 & 0 & 0 \\ 0 & 0 & \cdots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & 0 & \cdots & a_{n-1} & b_{n-1} & c_{n-1} & 0 \\ 0 & 0 & 0 & 0 & \cdots & a_n & b_n & c_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix}$$

Cyclic Reduction Algorithm

Row operations are carried out such that non-diagonal entries are moved further from the diagonal.

$$\begin{bmatrix} a_1^{(1)} & b_1^{(1)} & 0 & c_1^{(1)} & 0 & \cdots & 0 & 0 \\ 0 & a_2^{(1)} & I & c_2^{(1)} & 0 & \cdots & 0 & 0 \\ 0 & a_3^{(1)} & 0 & b_3^{(1)} & 0 & c_3^{(1)} & 0 & 0 \\ 0 & 0 & \cdots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & 0 & \cdots & a_{n-1}^{(1)} & I & c_{n-1}^{(1)} & 0 \\ 0 & 0 & 0 & \cdots & a_n^{(1)} & 0 & b_n^{(1)} & c_n^{(1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} f_1^{(1)} \\ f_2^{(1)} \\ f_3^{(1)} \\ \vdots \\ f_{n-1}^{(1)} \\ f_n^{(1)} \end{bmatrix}$$

Cyclic Reduction Algorithm

After $a = \text{ceil}\left(\log_2\left(n\right)\right)$ cycles:

$$\begin{bmatrix} a_1^{(a)} & b_1^{(a)} & 0 & 0 & 0 & \cdots & 0 & c_1^{(a)} \\ 0 & a_2^{(1)} & I & c_2^{(1)} & 0 & \cdots & 0 & 0 \\ 0 & a_3^{(2)} & 0 & I & 0 & c_3^{(2)} & 0 & 0 \\ 0 & 0 & \cdots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & 0 & \cdots & a_{n-1}^{(1)} & I & c_{n-1}^{(1)} & 0 \\ 0 & \cdots & a_n^{(a)} & \cdots & 0 & 0 & b_n^{(a)} & c_n^{(a)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} f_1^{(a)} \\ f_2^{(1)} \\ f_3^{(2)} \\ \vdots \\ f_{n-1}^{(1)} \\ f_n^{(a)} \end{bmatrix}$$

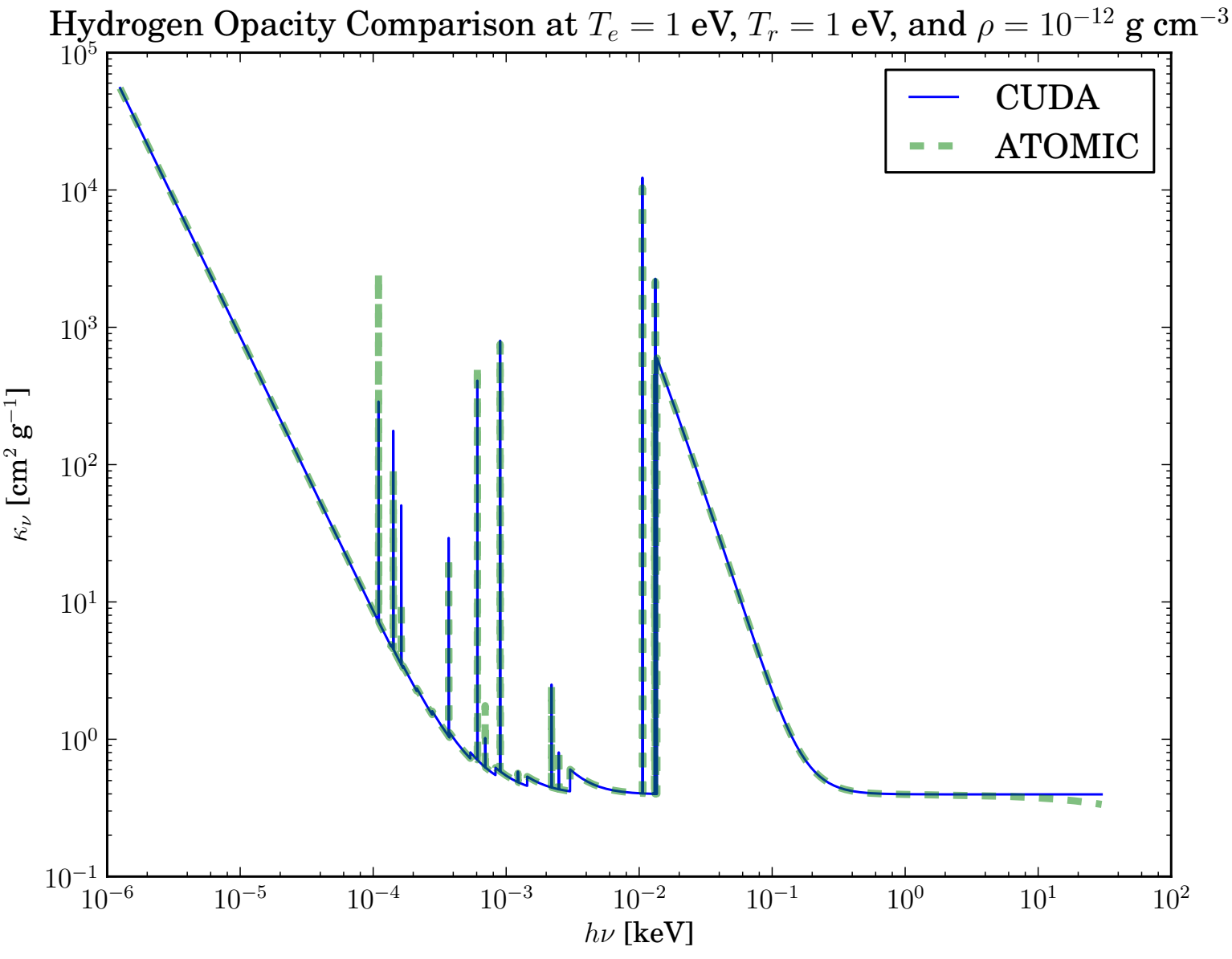
Solve for first block row unknowns and back-substitute.

Iterative Solution Method

- N_e is needed to compute matrix and rhs vector, iteratively solve for N_e.
- Compute all rate coefficients (eventually in parallel).
- Compute non-linear residual at physical limits of $N_e \in [q, ZN_{tot} + q]$ (requires 2 matrix solves).
- Compute matrix and rhs vector (eventually in parallel) and solve (eventually in parallel)
  - Matrix and RHS computation – atomic add
  - Matrix Solve – cyclic reduction
- Use conservation of particles constraint to compute the residual.
  - The function whose root we are finding is: $f(N_e) = \rho - \rho_{\text{calculated}}$
  - Solve iterative problem using Brent's method, a robust hybrid algorithm using inverse quadratic interpolation, the secant method, and bisection.

Methods of verification

- There are no known analytic test cases for these types of problems.
- This makes verification especially difficult.
- It was determined that comparisons with ATOMIC was the most tractable method of verification.
- The code described here and ATOMIC used identical rDCA atomic models.


Hydrogen Opacity Comparison at $T_e = 1$ eV, $T_r = 1$ eV, and $\rho = 10^{-12}$ g cm^{-3}
CUDA
ATOMIC
κ_ν [cm^2 g^{-1}]
$h\nu$ [keV]
10^{5}
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}

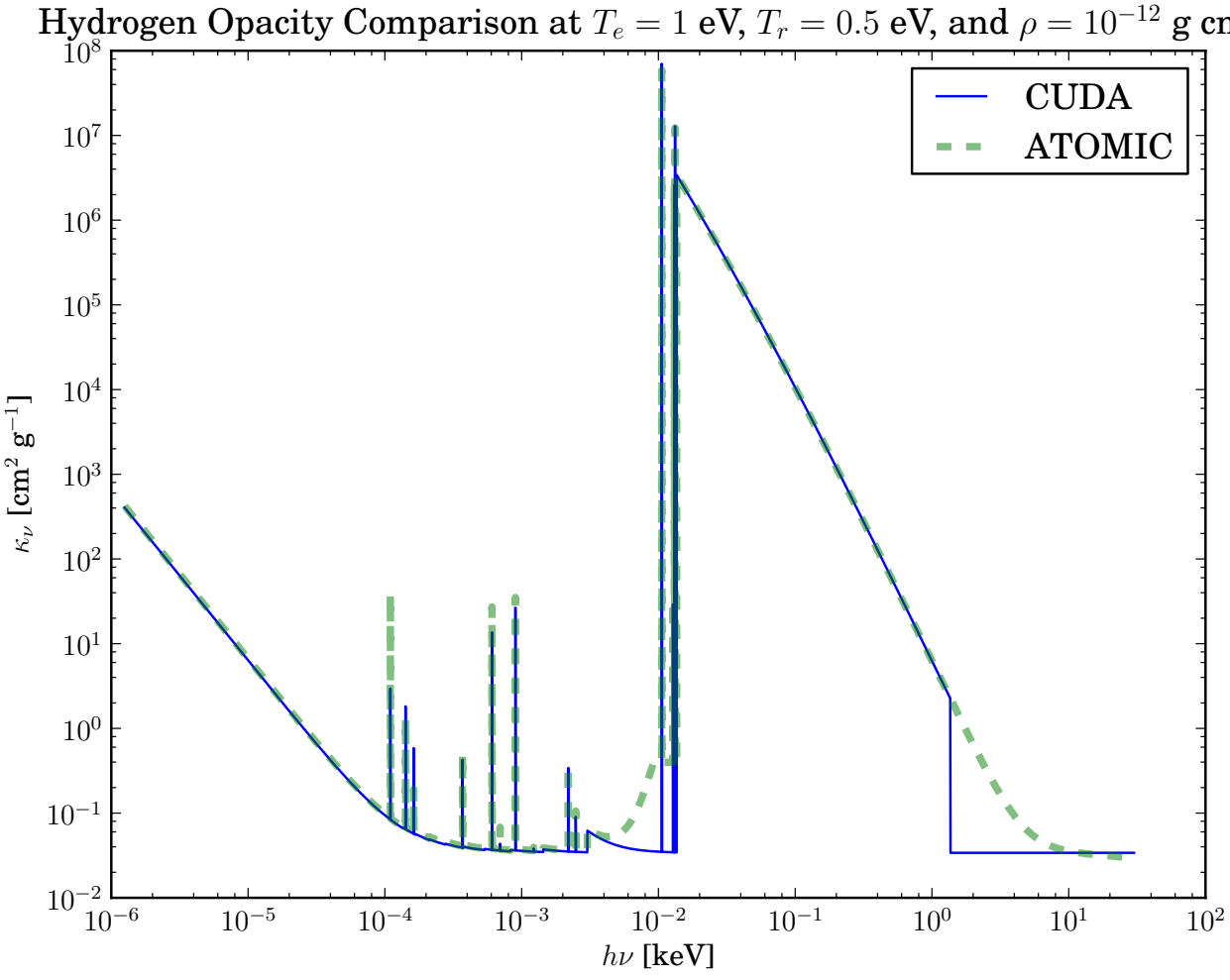


Figure : Some differences likely due to differences in scattering model (Thomson vs. a more detailed treatment) and differences in where fit functions turn off.

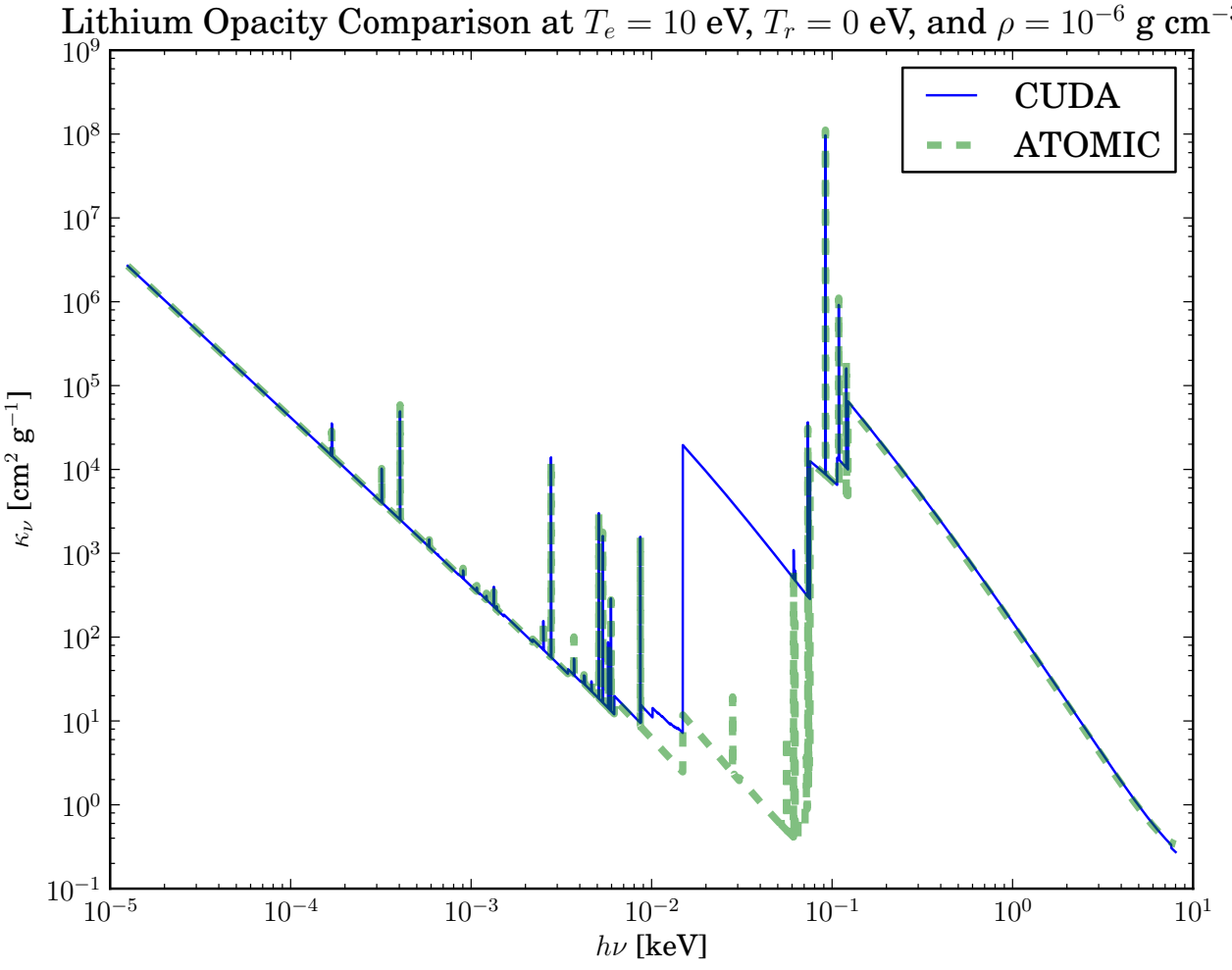


Figure : Orders of magnitude difference in one of the bound-free edges, reason is currently not known.

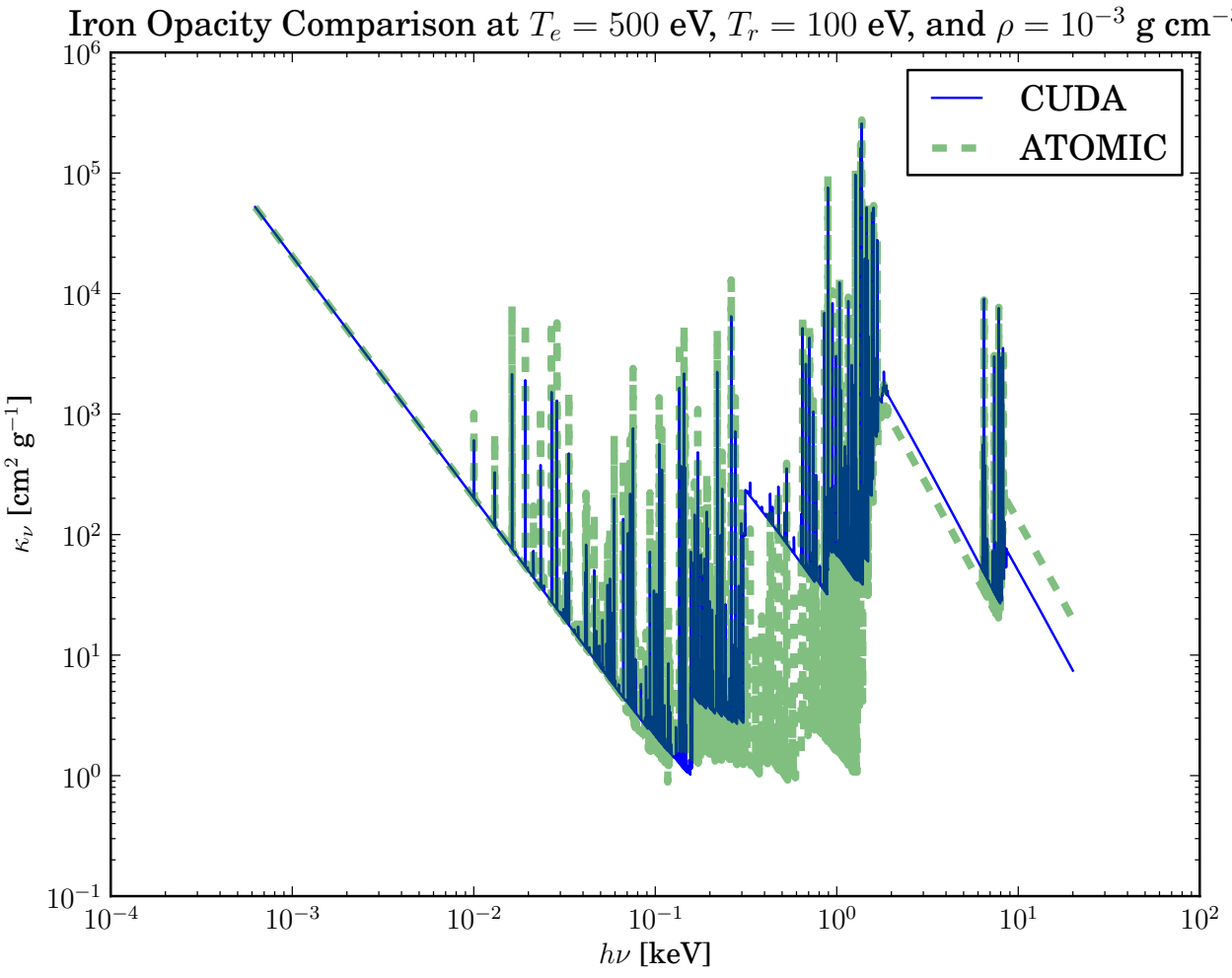


Figure : Orders of magnitude difference in one of the bound-free edges, reason is currently not known.

Comparisons of $\langle Z \rangle$

Model-Element	ρ [g/cc]	T_e [eV]	T_r [eV]	Code	$\langle Z \rangle$
rDCA Hydrogen	1e-08	2.0	0.5	ATOMIC	9.93068e-01
rDCA Hydrogen	1e-08	2.0	0.5	CUDA	9.93068e-01
DCA Hydrogen	1e-08	2.0	0.5	ATOMIC	9.93311e-01
DCA Hydrogen	1e-08	2.0	0.5	CUDA	9.93311e-01
rDCA Hydrogen	1e-08	2.0	2.0	ATOMIC	9.99047e-01
rDCA Hydrogen	1e-08	2.0	2.0	CUDA	9.99047e-01
DCA Hydrogen	1e-08	2.0	2.0	ATOMIC	9.99047e-01
DCA Hydrogen	1e-08	2.0	2.0	rDCA	9.99046e-01
DCA Lithium	1e-06	5.0	1.0	ATOMIC	1.01085e+00
DCA Lithium	1e-06	5.0	1.0	CUDA	1.01086e+00
DCA Lithium	1e-06	5.0	5.0	ATOMIC	1.25324e+00
DCA Lithium	1e-06	5.0	5.0	CUDA	1.25324e+00
DCA Carbon	1e-04	20	20	ATOMIC	3.98680e+00
DCA Carbon	1e-04	20	20	CUDA	3.98680e+00
rDCA Iron	0.001	500.0	100.0	ATOMIC	2.00614e+01
rDCA Iron	0.001	500.0	100.0	CUDA	2.00613e+01
rDCA Iron	0.001	500.0	500.0	ATOMIC	2.40049e+01
rDCA Iron	0.001	500.0	500.0	CUDA	2.40049e+01

Status

- Excellent indicators showing that the code can correctly compute $\langle Z \rangle$.
  - This is the most important aspect of this project.
  - xRage implementation only computes $\langle Z \rangle$ and uses that value in an LTE table lookup.
- Some issues with the opacity, but we are confident that we are close.
- Currently standalone and cannot handle multiple (T_e, T_r, ρ) states simultaneously.
  - Handling multiple pairs simultaneously is necessary for coupling to a rad-hydro code and exposes another level of parallelism.

Why CUDA?

- Want to accelerate the code, GPUs are one way to accelerate, CUDA is a simple way to use GPUs.
- Relatively lower development time when compared to OpenCL
- Easy access to cuBLAS library
- Excellent toolbox
  - Debugger that can debug CUDA kernels (cuda-gdb)
  - Visual profiler allows for kernel optimization (nvvp)
- Verified cyclic reduction solver written in CUDA using cuBLAS.
  - CUDA only works on NVIDIA GPUs (portability?)
  - cuBLAS does not have full support for partial pivoting with LU decomposition
- CUDA implementations for 60% of rate coefficient computation.
- cuBLAS very similar to MAGMA, which is a project aims to develop a dense linear algebra library similar to LAPACK but for heterogeneous/hybrid architectures, starting with current "Multicore+GPU" systems.
  - Nearly identical function calls to cuBLAS.
  - Portable to many different types of hardware.

Performance Comparison

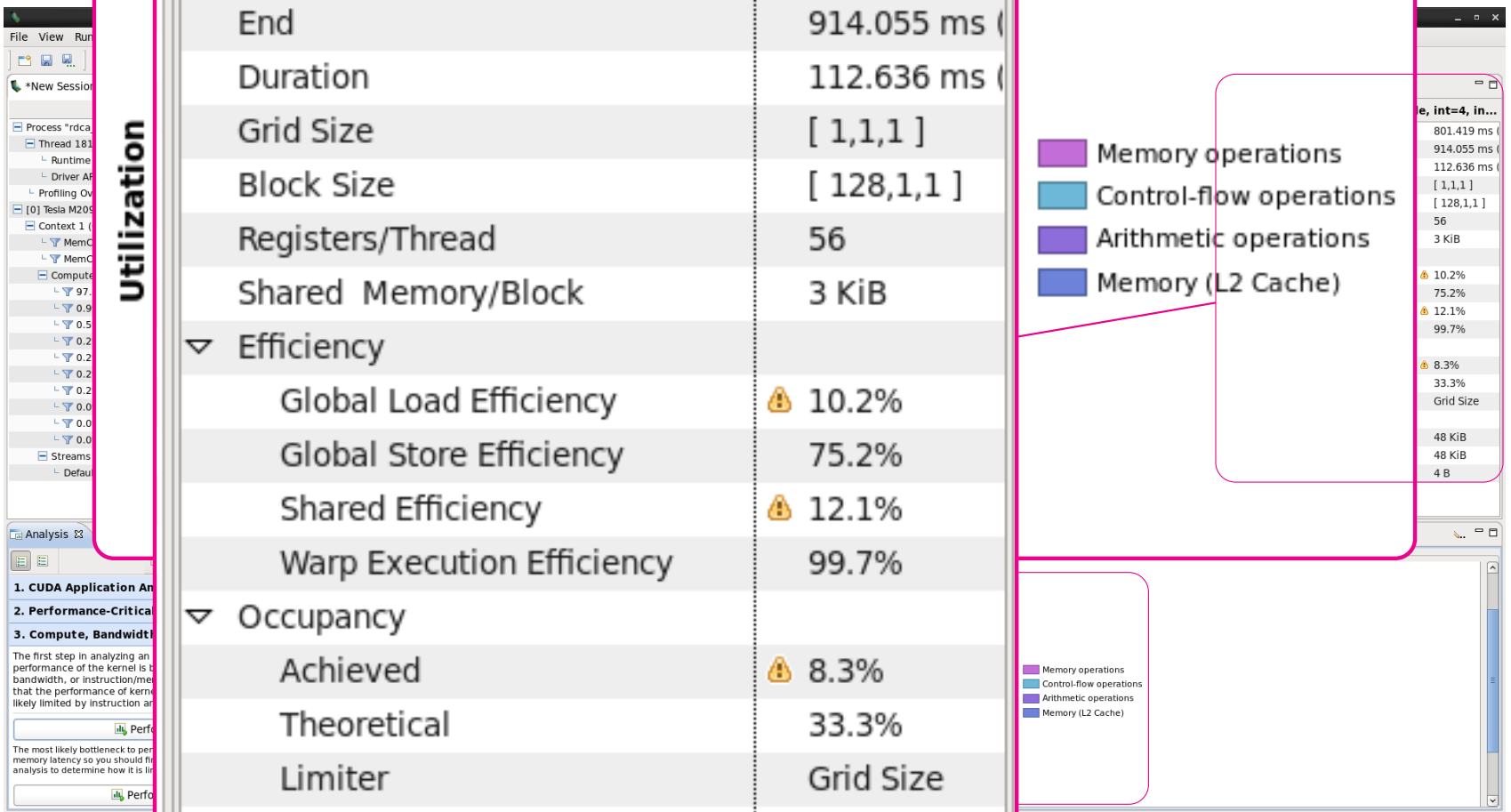
Model-Element	ρ [g/cc]	T_e [eV]	T_r [eV]	solver	CPU time [s]
rDCA Hydrogen	1e-08	2.0	0.5	ATOMIC	1.0e-02
rDCA Hydrogen	1e-08	2.0	0.5	CUDA	4.7e+00
DCA Hydrogen	1e-08	2.0	0.5	ATOMIC	2.6e-01
DCA Hydrogen	1e-08	2.0	0.5	CUDA	4.5e+00
rDCA Hydrogen	1e-08	2.0	2.0	ATOMIC	1.0e-02
rDCA Hydrogen	1e-08	2.0	2.0	CUDA	4.7e+00
DCA Hydrogen	1e-08	2.0	2.0	ATOMIC	0.0e+00
DCA Hydrogen	1e-08	2.0	2.0	CUDA	4.4e+00
DCA Lithium	1e-06	5.0	1.0	ATOMIC	2.4e+01
DCA Lithium	1e-06	5.0	1.0	CUDA	9.2e+00
DCA Lithium	1e-06	5.0	5.0	ATOMIC	2.3e+01
DCA Lithium	1e-06	5.0	5.0	CUDA	9.3e+00
DCA Carbon	1e-04	20	20	ATOMIC	1.3e+03
DCA Carbon	1e-04	20	20	CUDA	2.3e+02
rDCA Iron	0.001	500.0	100.0	ATOMIC	1.9e+00
rDCA Iron	0.001	500.0	100.0	CUDA	5.2e+00
rDCA Iron	0.001	500.0	500.0	ATOMIC	1.9e+00
rDCA Iron	0.001	500.0	500.0	CUDA	5.2e+00

The NVIDIA Visual Profiler

Memory operations account for much of total operations, also pointing to potentially poor co

Low global load effi-

r



Properties

void getrf_panel<double, double, int=4, in...

Start	801.419 ms (
End	914.055 ms (
Duration	112.636 ms (
Grid Size	[1,1,1]
Block Size	[128,1,1]
Registers/Thread	56
Shared Memory/Block	3 KiB
Efficiency	
Global Load Efficiency	10.2%
Global Store Efficiency	75.2%
Shared Efficiency	12.1%
Warp Execution Efficiency	99.7%
Occupancy	
Achieved	8.3%
Theoretical	33.3%
Limiter	Grid Size
Shared Memory Configuration	
Shared Memory Requested	48 KiB
Shared Memory Executed	48 KiB
Shared Memory Bank Size	4 B

Meeting the Goals

- Goal #1: To produce a robust code capable of computing reasonably accurate opacities with the assumption of LTE relaxed.
  - Succeeded in computing $\langle Z \rangle$
  - Mostly there for κ_ν
- Goal #2: Use efficient algorithms that expose high levels of parallelism and utilize good memory access patterns for use on advanced architectures.
  - Verified cyclic reduction solver using cuBLAS API function calls.
  - Exposed high levels of parallelism in rate coefficient computations via CUDA kernels.
- Goal #3: Portability to multiple types of hardware including multicore processors, graphics processing units (GPUs), and coprocessors such as the intel Xeon Phi
  - Eventually switch solver to MAGMA API function calls
  - Convert optimized CUDA kernels to OpenCL kernels
  - Code designed so that implementations can be swapped out, but "main" remains unchanged.
- Goal #4: Easily coupled to radiation hydrodynamics and thermal radiative transfer simulation environments.

Thanks for Listening!
Questions?

The Rate Coefficients

- Example: Total Radiative Ionization Rate (includes radiative recombination)

$$P_{i\ell} = \sum_{j=1}^{I} \sum_{m=1}^{L_j} p_{jm\to i\ell}(T_r) N_{jm} - r_{i\ell\to jm}(T_e) N_e N_{i\ell} - q_{i\ell\to jm}(T_e, T_r) N_e N_{i\ell}$$
$$- p_{i\ell\to jm}(T_r) N_{i\ell} + r_{jm\to i\ell}(T_e) N_e N_{jm} + q_{jm\to i\ell}(T_e, T_r) N_e N_{jm}$$

- For ionization coefficients like this one, we actually only consider events that change the ionicity by ± 1 as multiple ejection/recombination events are much less likely.
  - $j = i-1,\ i,\ i+1$, this implies a block tridiagonal structure.
- For excitation coefficients, $j = i$ only.
- Many $i\ell \to jm$ pairs not possible.
  - Quantum mechanical selection rules

The Rate Coefficients

Sub-Example: Radiative Recombination rate coefficient accounting for stimulated emission.

$$q_{jm\to i\ell}(T_e, T_r) = \frac{h^3 c}{\pi\sqrt{2^7 m_e^3}} \frac{g_{i\ell}}{g_{jm}} \int_0^{+\infty} \underbrace{F(E_e, T_e)}_{\text{Electron Energy Distribution}} E_e^{-1/2} \sigma_{jm\to i\ell}(E_{\text{ion pot.}} + E_e) \times \underbrace{G(E_{\text{ion pot.}} + E_e, T_r)}_{\text{Radiation Energy Distribution}} \, \mathrm{d}E_e$$

In the code:

- A Planckian radiation distribution is assumed unless otherwise specified
- A Maxwell-Boltzmann electron distribution is ALWAYS used.
  - Integrals only involving the electron distributions are pre-evaluated.
- Fit coefficients for the fundamental cross sections (σ's) are read in from a data file.
- Rate coefficient computation is an embarrassingly parallel task (no processor communication required).

Solving the Linear system

- We now have the tools to compute the rate matrix. We will get a system: $A(N_e)\mathbf{x} = \mathbf{0}$ and $|A| \neq 0$.
  - Problem: The solution is always the trivial solution $\mathbf{x} = \mathbf{0}$.
- If it is known that $\frac{\mathrm{d}N_{i\ell}}{\mathrm{d}t} = 0 \ \forall i, \ell$ except 1, then that last one must also be in equilibrium because it has no net sources or sinks.
- Solution: divide every equation by that last unknown and subtract the resulting constant over to the RHS and remove the row corresponding to the unknown we are dividing by.
- This will result in a modified matrix and unknowns vector reduced by 1 in size and whose unknowns are normalized by the "missing" unknown: $A'(N_e)\mathbf{x}' = \mathbf{b}(N_e)$, where $\mathbf{b}(N_e) \neq \mathbf{0}$.
- Take advantage of block tri-diagonal matrix structure in 2 ways:
  - Memory $\rightarrow$ efficient storage
  - Solvable in parallel via cyclic reduction

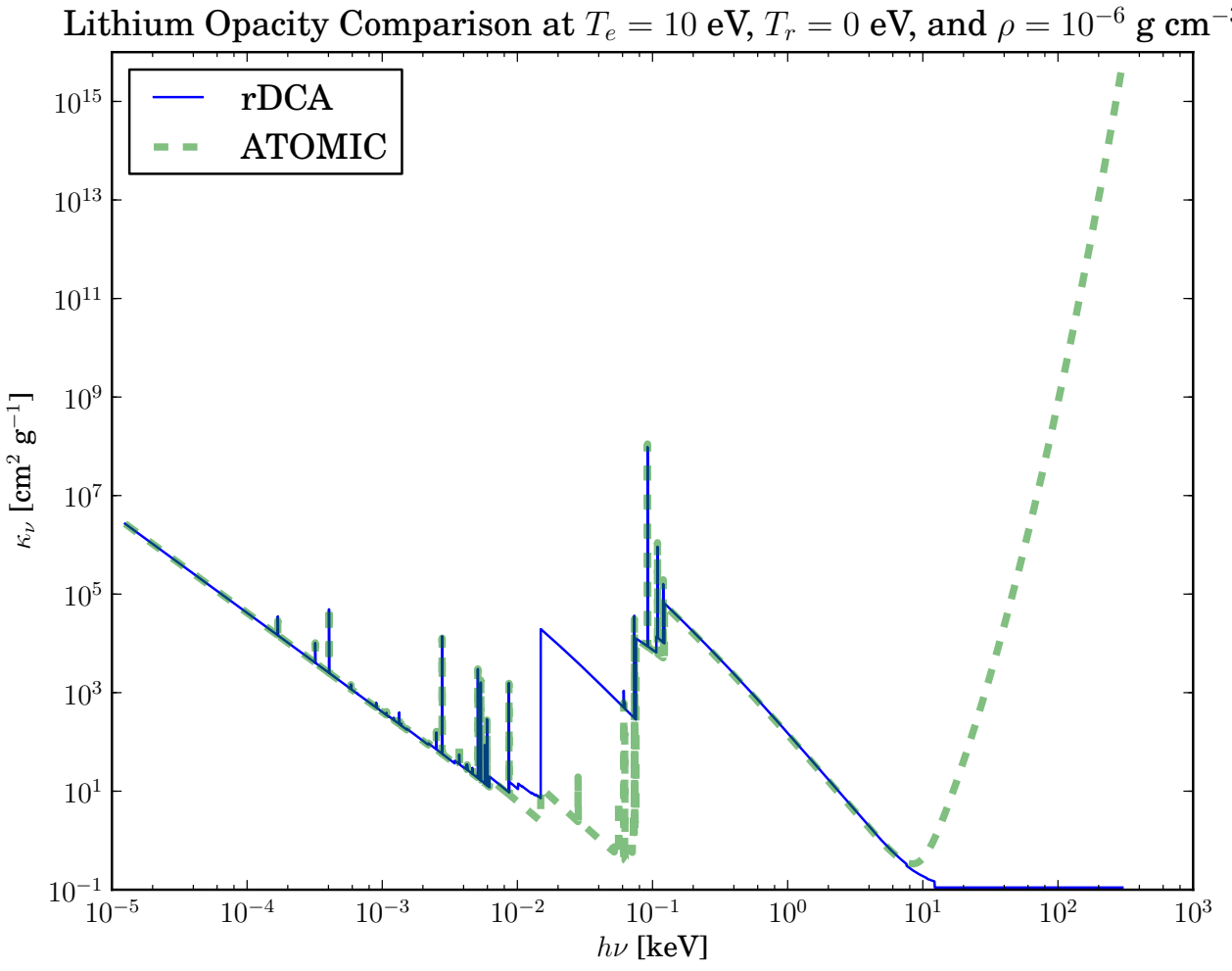


Figure : Orders of magnitude difference in one of the bound-free edges, reason is currently not known.

Beyond the Goals

- Computation of other material properties – emissivity, EOS corrections
- Use cyclic reduction and other parallel techniques directly in the ATOMIC suite
- Multiple materials